

## Assignment 2

### Problem 1: Parse Tree and Leftmost Derivation

Using the grammar below, show a parse tree and a leftmost derivation for the following statement:

$A = B * (C * (A + B))$

Grammar:

Statement  $\rightarrow$  Assignment | Expression

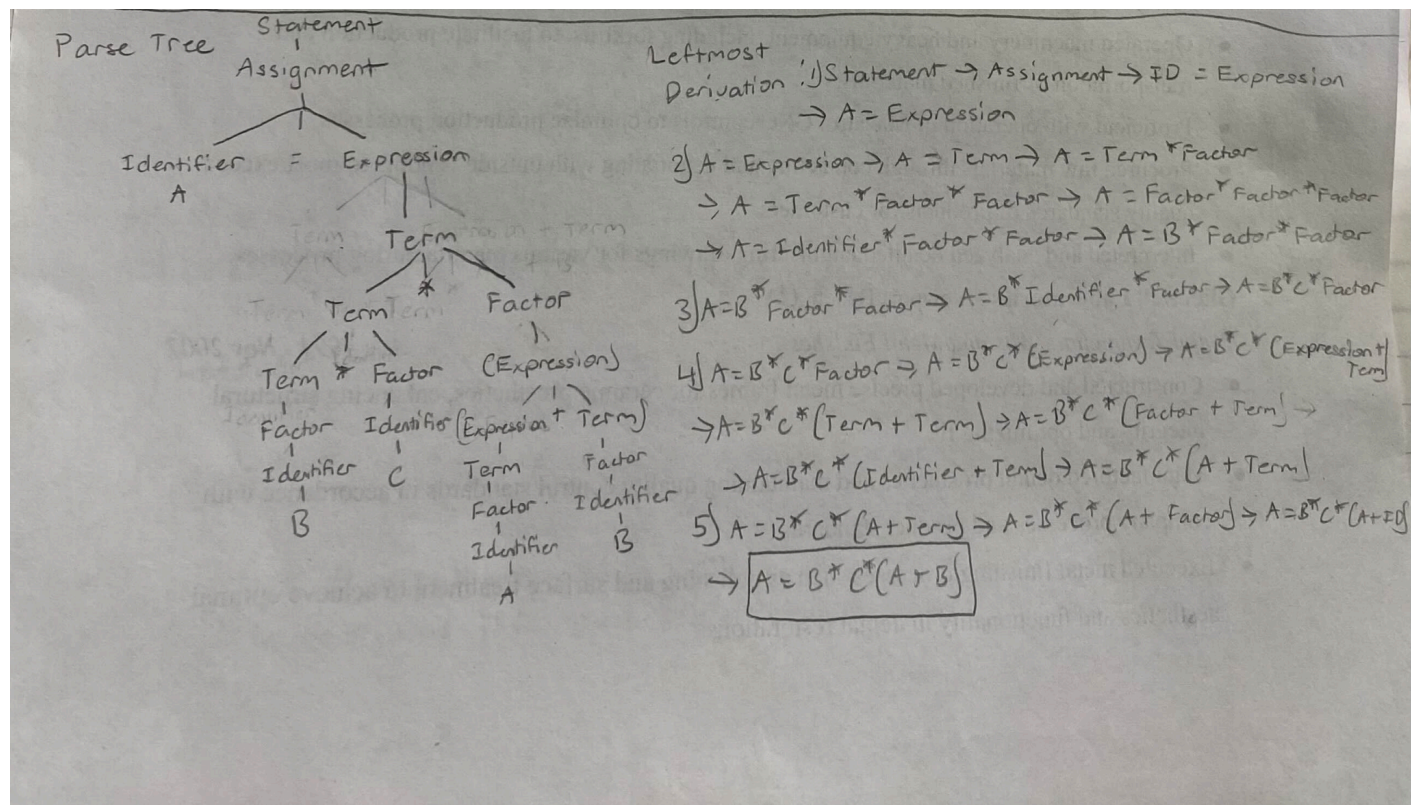
Assignment  $\rightarrow$  Identifier = Expression

Expression  $\rightarrow$  Expression + Term | Term

Term  $\rightarrow$  Term \* Factor | Factor

Factor  $\rightarrow$  ( Expression ) | Identifier

Identifier  $\rightarrow$  A | B | C



Leftmost Derivation (in text):

1) Statement  $\rightarrow$  Assignment  $\rightarrow$  Identifier = Expression  $\rightarrow$

**A = Expression**

2) A = Expression  $\rightarrow$  A = Term  $\rightarrow$  A = Term \* Factor  $\rightarrow$  A = Term \* Factor \* Factor  $\rightarrow$

A = Factor \* Factor \* Factor  $\rightarrow$  A = Identifier \* Factor \* Factor  $\rightarrow$

**A = B \* Factor \* Factor**

3) A = B \* Factor \* Factor  $\rightarrow$  A = B \* Identifier \* Factor  $\rightarrow$

**A = B \* C \* Factor**

4) A = B \* C \* Factor  $\rightarrow$  A = B \* C \* (Expression)  $\rightarrow$  A = B \* C \* (Expression + Term)

$\rightarrow$  A = B \* C \* (Term + Term)  $\rightarrow$  A = B \* C \* (Factor + Term)  $\rightarrow$  A = B \* C \*

(Identifier + Term)  $\rightarrow$

**A = B \* C \* (A + Term)**

5) A = B \* C \* (A + Term)  $\rightarrow$  A = B \* C \* (A + Factor)  $\rightarrow$  A = B \* C \* (A + Identifier)  $\rightarrow$

**A = B \* C \* (A + B)**

---

Problem 2: Scope Concepts

Considering the following program written in pseudocode:

```
int u = 42;
```

```
int v = 69;
```

```
int w = 17;
```

```
proc add( z:int )
```

```
  u := v + u + z;
```

```
proc bar( fun:proc )
```

```
  int u := w;
```

```
  fun(v);
```

```

proc foo( x:int, w:int )
int v := x;
bar(add);

```

```

main
foo(u, 13);
print(u);
end;

```

a. Using Static Scope, what is printed to the screen?

Static: not looking down at the stack but takes global values

add(v)	
bar	v = 69
foo(u, 13)	v = 69, u = 42, w = 17
main	u, v, w, x

add(v)

u:  $v + u + v = 69 + 42 + 69 = 180$

- In Static Scope, the program would print out '180' since it loses reference to the local variable 'u' after the add/bar functions.

Therefore, the program  $u = (v + u + v) = (69 + 42 + 69) = \underline{180}$ .

b. Using Dynamic Scope with Deep Binding, what is printed to the screen?

Hint: The sum for u is 126, but due to deep binding, it's foo's local v that gets involved.

Dynamic deep

add(v)	
--------	--

bar	v = 42, u = 42
foo(u, 13)	v = 42, u = 42, w = 17
main	u = 42 v = 69 w = 17 x = 42

add(v)

- Deep binding would pass add into foo where u = 42, v = 42, and w = 17. Then, the function is passed into bar where v = 42 and u = 42.

Therefore, the localized variable v is equal to 42 instead of 69, and u is still equal to 42.

u:  $v + u + v = 42 + 42 + 42 = \underline{126}$

c. Using Dynamic Scope with Shallow Binding, what is printed to the screen?

Hint: The sum for u is 101, but again it's foo's local v that matters.

Dynamic shallow

add(v)	
bar	v = 42, u = 13
foo(u, 13)	v = 42, u = 42, w = 13
main	u = 42 v = 69 w = 17 x = 42

add(v)

- Shallow binding would pass add into foo where  $u = 42$ ,  $v = 42$ , but  $w = 13$ . Then, the function is passed into bar where  $v = 42$  and  $u = 13$  instead of 42.

$$u: v + u + v = 42 + 13 + 42 = 97$$

However, the program still views the global variable  $u$  as 42 despite the function call foo( $u$ , 13), so the output of  $u$  would be **42**.